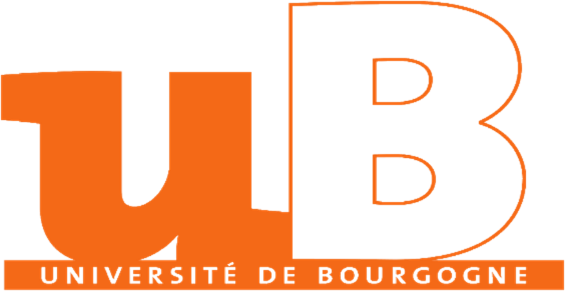


Javafft

LACHAUD Samuel / PAZOLA Loïs – Info S5 TP4



Une image contenant texte

Description générée automatiquement

2/7

I

II

IV

III

SOMMAIRE



Introduction



3/7

Afin de mener à bien le projet d’outils mathématiques, nous avons choisis d’utiliser le langage JAVA.

Cependant dans JAVA, il n’existe pas de bibliothèques pour la gestion des nombres complexes. Ainsi nous avons donc créé une classe **Complexe** dans notre projet.

La classe **Complexe** est composée de deux attributs :

1. double real = 0; // désigne la partie Réelle du complexe.
2. Double imag = 0; // désigne la partie Imaginaire du complexe.

Nous avons ensuite ajouté quelques opérations afin de pouvoir effectuer les calculs élémentaires :

1. Complexe add(Complexe c) // ajout d’un complexe c à celui que l’on a.
2. Complexe minus(Complexe c) // retrait d’un complexe c à celui que l’on a.
3. Complexe multiply(Complexe c) // multiplie le complexe que l’on a par c.
4. Complexe divide(Complexe c) // divise le complexe que l’on a par c.

Transformée 1D



4/7

avec

avec u = 0..N-1.

avec x = 0..N-1.

Notre méthode de transformée 1D est composée d’un tableau en une dimension qui stocke des nombres complexes précédemment vus. Nous avons également un paramètre sensTransformée :

* Si sensTransformee vaux -1 alors nous sommes en train de faire une Transformée de Fourier 1D.
* Si sensTransformee vaux +1 alors nous sommes en train de faire la Transformée de Fourier 1D Inverse.

Voici la formule de la transformée 1D et de l’inverse :

Nous pouvons donc voir que la différence entre ces deux formules sont le signe du -2 dans l’exp() et le 1/N qui multiplie la somme.

Nous avons donc décidé de mettre la transformée inverse directement dans la même méthode que la transformée 1D.

La somme est représentée par une boucle for avec j allant de 0 à N – 1, de la même manière, la deuxième boucle for représente le parcours du tableau de complexes. :

1. for(int j = 0 ; j < N ; j++) {//code }

Puisque nous utilisons la classe Complexe, nous avons besoin de décomposer la formule suivante :

* La partie réelle du complexe est donc cos(θ) \* ReelleTableau
* La partie imaginaire sin(θ) \* ImaginaireTableau.

Enfin, pour chacune des cases du tableau, si SensTransformee est dans le sens inverse, on divise la somme à mettre dans la case par la taille du tableau N (pareil que de multiplier par 1/N).



Transformée Rapide 1D

5/7



6/7

Notre méthode de transformée 1D Rapide est composée d’un tableau en une dimension qui stocke des nombres complexes précédemment vus. Nous avons également un paramètre sensTransformée comme la transformée 1D classique. Voici la formule

* TransformeeRapide1D 🡪 met en place le tableau et les données pour appel du calcul.
* InversionTableau 🡪 Inverse le sens du tableau (on le lis du dernier index au premier).
* CalculTransformeeRapide1D 🡪 effectue récursivement les calculs de la transformée rapide.

**TransformeeRapide1D :**

La transformée rapide 1D ne peut s’effectuer (facilement) qu’avec des tailles de tableau de taille 2^n. Nous faisons alors une comparaison bit à bit entre N et N-1, si cette comparaison renvoie autre chose que 0, alors nous ne sommes pas sur une puissance de 2 et on lève une Exception :

1. if( (N & (N - 1)) != 0)
2. {
3. throw new Exception("Le tableau doit avoir comme taille une puissance de 2 !");
4. }

Nous pouvons découper la formule en deux parties, la première partie à gauche du « + » désigne la partie avec les rangs paire du tableau (en rouge), la seconde partie désigne les rangs impairs du tableau (en violet). On effectuera donc sur chaque branche (impaire et paire) les calculs. Ceux-ci sont récursifs, on appellera donc la méthode CalculTransformeeRapide1D dans elle-même.

1. for(int i = 0; i < tableauR1D.size(); i++) {
2. if (i % 2 != 0) { // Si rang impair
3. part1.set(i / 2, tableauR1D.get(i));
4. }
5. else { // Si rang pair
6. part2.set((i + 1 ) / 2, tableauR1D.get(i));
7. }
8. }
9. ArrayList<Complexe> resPart1 = CalculTransformeeRapide1D(part1, SensTransformee);
10. ArrayList<Complexe> resPart2 = CalculTransformeeRapide1D(part2, SensTransformee);

**CalculTransformeeRapide1D :**

Pour voir le fonctionnement, nous allons faire l’algorithme à la main avec la formule suivante :

un tableau de taille **N = 4**. Prenons un tableau 1D avec pour valeurs **[1,3,2,0]**. **n** est la valeur qui parcours le tableau du rang 0 à N -1. On prend alors v comme variable pour la formule. On peut donc déterminer les sommes à la main :

1. X(n=0) = tf(0) + tf(1) + tf(2) + tf(3)
2. X(1) = tf(0) + tf(1)\***v1** + tf(2)\***v2** + tf(3)\***v3**
3. X(2) = tf(0) + tf(1)\***v2** + tf(2)\***v4** + tf(3)\***v6**
4. X(3) = tf(0) + tf(1)\***v3** + tf(2)\***v6** + tf(3)\***v9**

Le pas entre les différents exposants de w correspond avec la valeur n de X(n).

Une image contenant texte

Description générée automatiquement

7/7

Sources utilisées

On applique alors la Transformée de Fourier sur les rangs paires et impaires ce qui nous donne :



Effsd

La transformée 2D met en place la même formule que la 1D, mais avec deux boucles supplémentaires afin de couvrir la deuxième dimension du tableau et la deuxième dimension de calcul.

<https://perso.esiee.fr/~bercherj/New/polys/poly_tfd.pdf> 🡪 Cours de transformée d’une école d’ingénieur.  
<http://adrian.gaudebert.fr/downloads/licence3/maths/rapport-despret-gaudebert.pdf> 🡪 Un élève ayant travaillé sur un projet similaire  
<http://jl.baril.u-bourgogne.fr/c2.pdf> 🡪 Le cours de la faculté sur les transformée de Fourier

Transformée 2D